Porting code to OSG

Derek Weitzel

Structure

- Combining a few sections.
- Will first talk about high level porting, then give an example.
- After my short talk, will be experts time. OSG ‘experts’ will walk around and help users get running and answer questions.

Checklist for jobs

- Executable portable(ness)?
- Job Length
- Workflow Structure
- Data

Executable Portable(ness)

- Hardcoded paths will not be portable.
- Compile time file paths are also not portable.
- Required libraries. Static compiled applications are best, but can bring some libraries.

Subtle Executable Portable(ness)

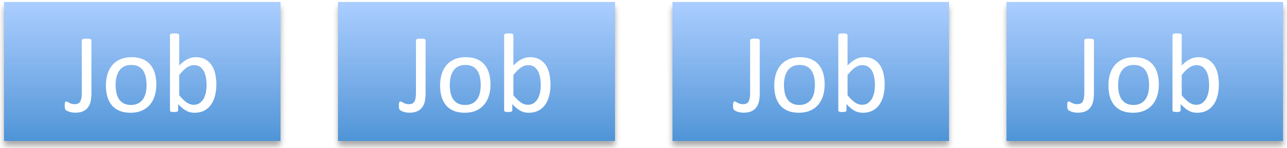
- File path length limits.
- Compiled with processor extensions (Intel vs AMD)
- Process forking – Scheduler can lose track of jobs.

Job Length

- Target job length is 1-3 hours.
- Bundle smaller jobs into a larger jobs. Usually easy.
- Division of larger jobs into smaller

Workflow Structure

- Flat structure?
  - Easy to work with.



Job

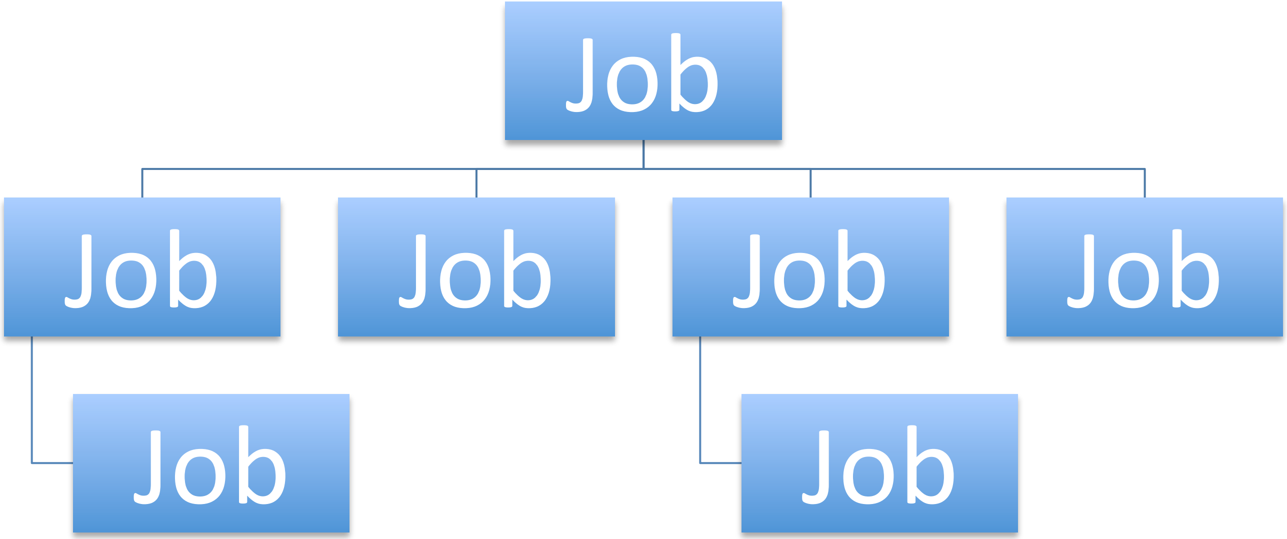
Job

Job

Job

Workflow Structure

- Hierarchal? (DAG)
  - – Can create complex workflows that automate retries and job execution.



Data

- Pull
  - Worker node pulls in data at beginning of execution
  - Late binding attributes that fits well with Pilot based workflows (Don't know where job will end up)

- Push
  - Push the jobs to the data
  - Accomplished with pre-staging

Data - Pull

- Common way is through HTTP Squid Caching
  - Documented on twiki https://twiki.grid.iu.edu/bin/view/Documentation/OsgHttpBasics

- SRM Transfers
  - SRM copy for larger files. SRM accesses larger storage elements that have high bandwidth and high capacity.

Data - Push

- Push to sites
  - Push to a local storage element ‘near’ the compute element
  - Usually done by automated infrastructure.
  - Limits you to running where your data is, even though those sites may be full

Data - Applications

- Does my application need access to all of this global data?

- If it only needs a small portion, then only transfer small portion.

- Don't use the global storage space, OSG_APP & OSG_DATA (unless you have to)

HCC’s example

- Open Mass Spectrometry Search Algorithm (OMSSA) from Nebraska Medical Center
- 22,000+ (short) Jobs per dataset, divided into ~130 jobs per real job, 172 runs per dataset
- x45 Data sets = 961200 (short) jobs

HCC’s example

- 21MB for Per Dataset shared between datasets

- 83MB for Executables. Used in every job

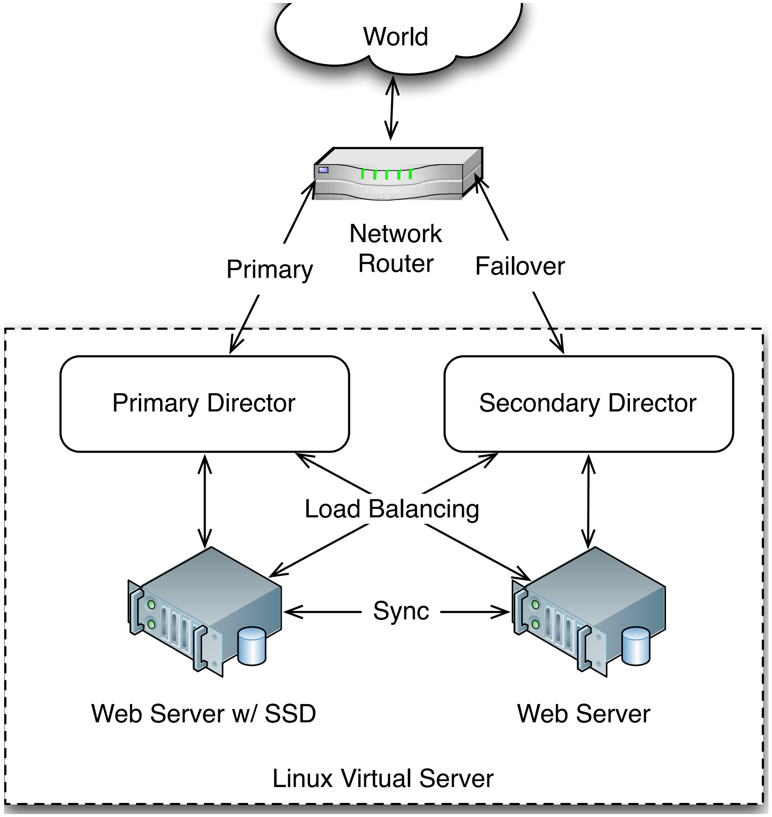
- 172 Runs per Dataset

HCC’s Example – Data

- Decided to use Squid
  - CMS and ATLAS sites required to have Squid
- Both executables (shared by all runs) and datasets (shared by subset jobs) will be cached.
- Made special Linux Virtual Server at HCC

HCC’s Example – Data

- Made special Linux Virtual Server at HCC
  - Handles high load from sites without squid.



11 requests/sec - 0.6 GB/second - 58.2 MB/ request

Possible Questions

- Data questions?
  - How should I distribute Data?
  - How much data can Squid handle?

- I have a rule of thumb:
  - 1-10 MB: Transfer data with each job
  - 10-200MB: Use squid
  - 200MB – 2-3GB: Use SRM
  - 3GB+: Special case. Need some clever thinking.

Possible Questions

- Workflow questions?
  - Use DAG?
    - Simple to understand and setup.
  - Use Pegasus?
    - Can handle complex (data centric) workflows
    - Growing to handle pilot submissions. Not 100% compatible with pilot systems yet.

Possible Questions

- Job questions?
  - What sites should I run on?
    - Can query to see sites that support your VO.
  - Should I use GlideinWMS (usually yes)
    - Most larger VO's use GlideinWMS.
  - What VO should I use?
    - If you are related to any existing VO's, talk to them.
    - Engage is a grab bag VO able to support many different sciences.